## IN THE CLAIMS

Please amend the claims as follows:

1-30. (Canceled)

31. (Currently Amended) A system comprising:

a random access memory to store at least a part of a number of processor instructions; and

a processor including, ~~to have a number of microarchitectural states during execution of the number of processor instructions, the processor comprising:~~

a ~~main~~first pipeline to execute the number of processor instructions;

a conjugate mapping table to store at least one entry that includes a trigger and an associated target related to execution of the number of processor instructions; and

~~an h-flow~~ a second pipeline to execute ~~h-flow~~ code related to the associated target if the trigger is satisfied during execution of the number of processor instructions, ~~wherein~~ the ~~h-flow~~ second pipeline [[is]] to modify one of ~~the~~ a number of ~~microarchitectural~~ states of the processor based on execution of the ~~h-flow~~ code.

32. (Currently Amended) The system of claim 31, wherein the processor further includes ~~comprises~~ a dynamic code analysis block to generate the ~~h-flow~~ code based on an analysis of the execution of the number of processor instructions by the main pipeline.

33. (Currently Amended) The system of claim 31, wherein the processor further includes ~~comprises~~ a ~~microarchitectural~~ structure to store the number of ~~microarchitectural~~ processor states.

34. (Currently Amended) The system of claim 31, wherein the number of processor ~~microarchitectural~~ states include values stored in register banks, branch target buffers or data cache.

35. (Currently Amended) A method comprising:

executing, in a first pipeline, a number of processor instructions;

storing, in a conjugate mapping table, at least one entry that includes a trigger and an associated target related to the execution of the number of processor instructions; and

executing, in ~~an h-flow~~ a second pipeline, ~~h-flow~~ code related to the associated target if the trigger is satisfied during the execution of the number of processor instructions, wherein the ~~h-flow~~ second pipeline is to modify one of ~~the~~ a number of ~~microarchitectural~~ processor states based on execution of the ~~h-flow~~ code.

36. (Currently Amended) The method of claim 35 further comprising:

generating the ~~h-flow~~ code based on an analysis of the execution of the number of processor instructions by the main pipeline.

37. (Currently Amended) The method of claim 35 further comprising:

storing the number of ~~microarchitectural~~ processor states.

38. (Currently Amended) The method of claim 35, wherein the number of ~~microarchitectural~~ processor states include values stored in register banks, branch target buffers or data cache.

39. (Previously presented) The method of claim 35, wherein the trigger is selected from a group consisting of instruction triggers, data triggers, state triggers and event triggers.

40. (Previously presented) The method of claim 35, wherein the trigger can include single atomic attributes or can include vector triggers.

41. (Currently Amended) A computer-readable medium containing computer instructions which when executed will perform the following:

executing, in a first pipeline, a number of processor instructions;

storing, in a conjugate mapping table, at least one entry that includes a trigger and an associated target related to the execution of the number of processor instructions; and

executing, in ~~an h-flow~~ a second pipeline, ~~h-flow~~ code related to the associated target if the trigger is satisfied during the execution of the number of processor instructions, wherein the ~~h-flow~~ second pipeline is to modify one of ~~the~~ a number of ~~microarchitectural~~ processor states based on execution of the ~~h-flow~~ code.

42. (Currently Amended) The computer-readable medium of claim 41 further containing computer instructions which when executed will perform:

generating the ~~h-flow~~ code based on an analysis of the execution of the number of processor instructions by the main pipeline.

43. (Currently Amended) The computer-readable medium of claim 41 ~~futher~~ further containing computer instructions which when executed will perform:

storing the number of ~~microarchitectural~~ processor states.

44. (Currently Amended) The computer-readable medium of claim 41, wherein the number of ~~microarchitectural~~ processor states include values stored in register banks, branch target buffers or data cache.

45. (Previously presented) The computer-readable medium of claim 41, wherein the trigger is selected from a group consisting of instruction triggers, data triggers, state triggers and event triggers.

46. (Previously presented) The computer-readable medium of claim 41, wherein the trigger can include single atomic attributes or can include vector triggers.

47. (New) A processor comprising:

a first pipeline to execute a number of processor instructions;

a table to store at least one entry that includes a trigger and an associated target related to execution of the number of processor instructions; and

a second pipeline to execute code related to the associated target if the trigger is satisfied during execution of the number of processor instructions, the second pipeline to modify one of a number of processor states based on execution of the code.

48. (New) The processor of claim 47, wherein the trigger is to check a location of one of the processor instructions to determine a potentially reusable processor instruction.

49. (New) The processor of claim 47, wherein the code is generated as a result of identifying ones of the number of processor instructions that are candidates for computation reuse, wherein the identifying includes creating an execution trace of ones of the number of processor instructions, compressing the execution trace to find recurrent portions thereof, and identifying the recurrent portions of the execution trace as reusable computation units.

50. (New) The processor of claim 49, wherein the creating the execution trace includes executing ones of the number of processor instructions and mapping architectural states of the ones of the number of processor instructions into symbols.

51. (New) The method of claim 36, wherein the analysis includes,

mapping n-dimensional architectural state vectors, which are representative of instances of ones of the number of processor instructions, into a plurality of one-dimensional symbols;

arranging the plurality of one-dimensional symbols into phrases of text; and

identifying recurrent phrases of text as reusable computation units.

52. (New) The method of claim 51, wherein the mapping includes,

traversing a software block in program execution order;

assigning new symbols as previously un-encountered architectural state vectors are encountered; and

assigning previously assigned symbols as previously encountered architectural state vectors are encountered.

53. (New) The method of claim 52, wherein assigning new symbols includes assigning consecutive integers such that each new symbol is assigned a value that is one greater than a previously assigned value.

54. (New) The method of claim 51, wherein the arranging includes arranging ones of the plurality of one-dimensional symbols in program execution order.

55. (New) The method of claim 51, wherein the architectural state vectors include live-in states and live-out states for ones of the processor instructions.

56. (New) The method of claim 51, wherein the identifying includes compressing the phrases of text to find a plurality of recurrent phrases.

57. (New) The method of claim 56, wherein the compressing includes compressing the phrases of text using a lossless compression algorithm.

58. (New) The method of claim 57 further comprising generating at least one trigger for a conjugate processor, the at least one trigger to implement complete reuse.

59. (New) The method of claim 56, wherein the compressing comprises compressing the phrases of text using a lossy algorithm.

60. (New) The method of claim 59 further comprising generating at least one trigger for a conjugate processor, the at least one trigger to implement partial reuse.

61. (New) The method of claim 56, wherein the identifying further includes correlating the plurality of recurrent phrases to identify reusable computation units.